嵌入式系统

Embedded System

毛维杰

杭州 • 浙江大学 • **2021**

第三章　MCS-51指令系统

- §3-1 指令系统概述
- §3-2 寻址方式
- §3-3 8051指令集

指令系统概述

MCS-51指令分类

MCS-51单片机共有111条指令。

1. 按指令所占的字节数分类

①单字节指令49条

②双字节指令46条

③三字节指令16条

2. 按指令执行时间长短分

①单周期指令64条

②双周期指令45条

③四周期指令2条

MCS-51指令分类（2）

3．按功能分为以下五种：　　C语言

①数据传送类指令(29条) ⟷ 赋值语句

②数据运算类指令(24条) ⟷ 算术运算

③逻辑操作类指令(24条) ⟷ 逻辑运算

④控制转移类指令(17条) ⟷ 选择和循环

⑤布尔操作类指令(17条) ⟷ 位运算

指令格式

[标号：]操作码　操作数1，操作数2[；注释]

例：　LOOP：　MOV　　A，#40H　；取参数

1. 标号：指令的符号地址
2. 操作码：指明指令功能
3. 操作数：指令操作对象、数据、地址、寄存器名及约定符号。

   操作数1：目的操作数，操作数2：源操作数
4. 注释：说明指令在程序中的作用。

操作码和操作数是指令主体，[]不是必须内容；

换行表示一条指令结束；

注意标点符号的使用。

指令描述符号介绍

- Rn —当前选中的寄存器区中的8个工作寄存器R0～R7（n=0～7）。
- Ri —当前选中的寄存器区中的2个工作寄存器R0、R1（i=0，1）。
- direct —8位的内部数据存储器单元中的地址。
- #data —包含在指令中的8位常数。
- #data16 —包含在指令中的16位常数。
- addr16 —16位目的地址。
- addr11 —11位目的地址。

指令描述符号介绍（2）

- rel —8位带符号的偏移字节，简称偏移量。
- DPTR —数据指针，可用作16位地址寄存器。
- bit —内部RAM或专用寄存器中的直接寻址位。
- A —累加器。
- B —专用寄存器，用于乘法和除法指令中。
- C —进位标志或进位位，或布尔处理机中的累加器。

指令描述符号介绍（3）

- @ ——间址寄存器或基址寄存器的前缀，如@Ri，@DPTR。
- / ——位操作数的前缀，表示对该位操作数取反，如/bit。
- × ——片内RAM的直接地址或寄存器。
- (×) ——由×寻址的单元中的内容。
- ← ——箭头左边的内容被箭头右边的内容所代替。

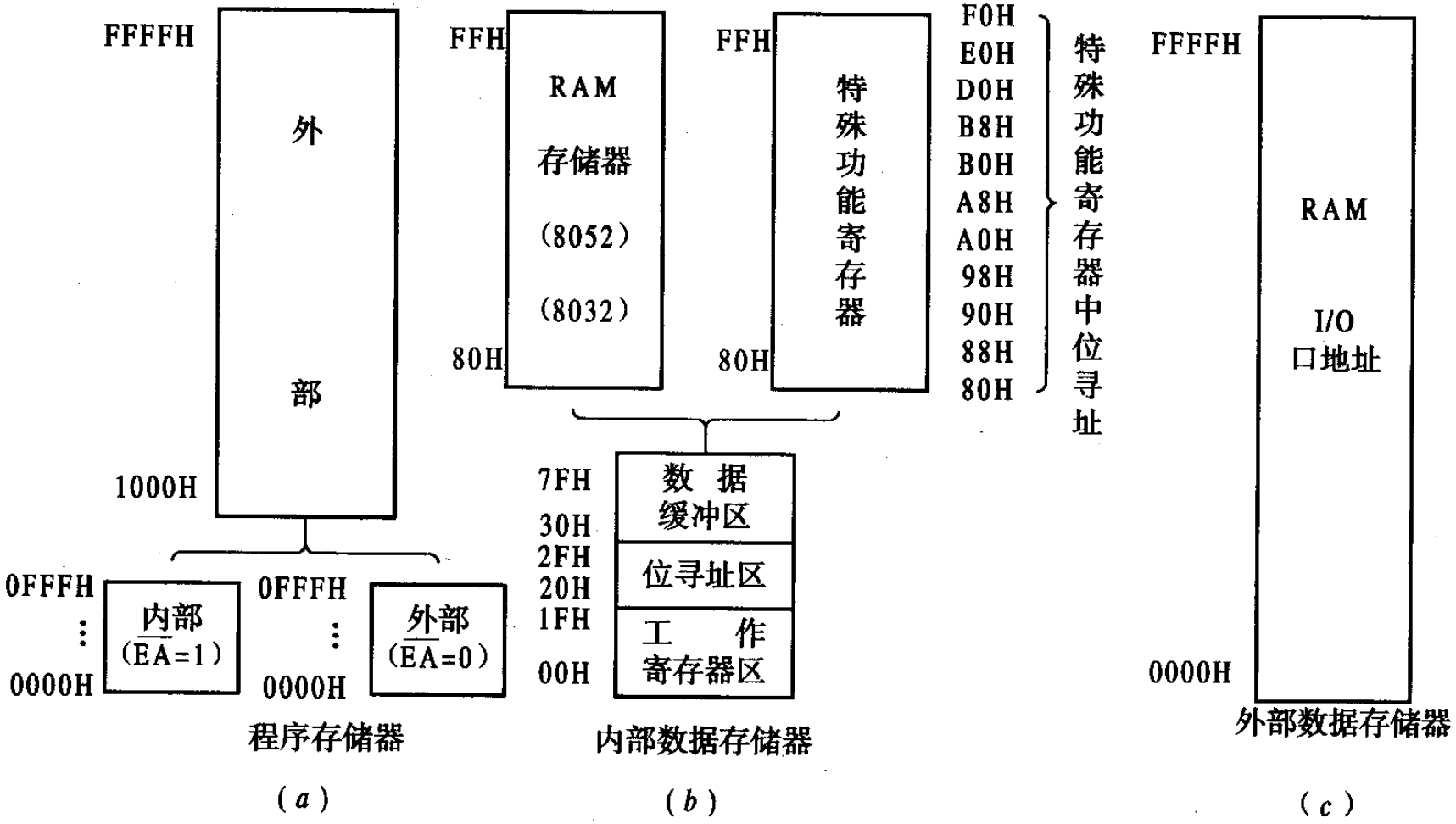
第三章　MCS-51指令系统

- §3-1　指令系统概述
- §3-2　寻址方式
- §3-3　8051指令集

寻址方式

- 寻址方式：寻找（或确定）操作数所在单元地址的方式。
- 在用汇编语言编程时，数据的存放、传送、 运算都要通过指令来完成。 编程者必须自始至终都要十分清楚操作数的位置，以及如何将它们传送到适当的寄存器去参与运算。
- 寻址方式越多，计算机寻址能力越强，但指令系统也越复杂。

MCS-51寄存器配置（物理空间）

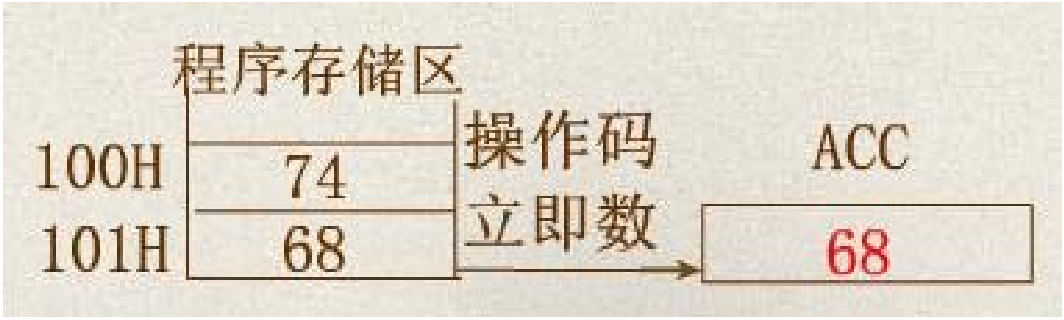


寻址方式:

- （1） 立即寻址;
- （2） 直接寻址;
- （3） 寄存器寻址;
- （4） 寄存器间接寻址;
- （5） 基址寄存器加变址寄存器间接寻址;
- （6） 相对寻址;
- （7） 位寻址。

立即寻址

- 立即寻址方式是指操作数包含在指令字节中。 跟在指令操作码后面的数就是参加运算的数，该操作数称为立即数。
- 立即数有一字节和二字节两种可能，例如指令:
  - MOV　A, ＃68H（74H，68H）
  - MOV　DPTR, ＃0DFFFH



直接寻址

- 在指令中直接给出操作数的地址，这种寻址方式就属于直接寻址方式。在这种方式中，指令的操作数部分直接是操作数的地址

- MOV A，3AH ；把片内RAM字节地址3AH单元的内容送累加器A中。

  MOV 3AH，A ；把A的内容传送给片内RAM的3AH单元中。

直接寻址（2）

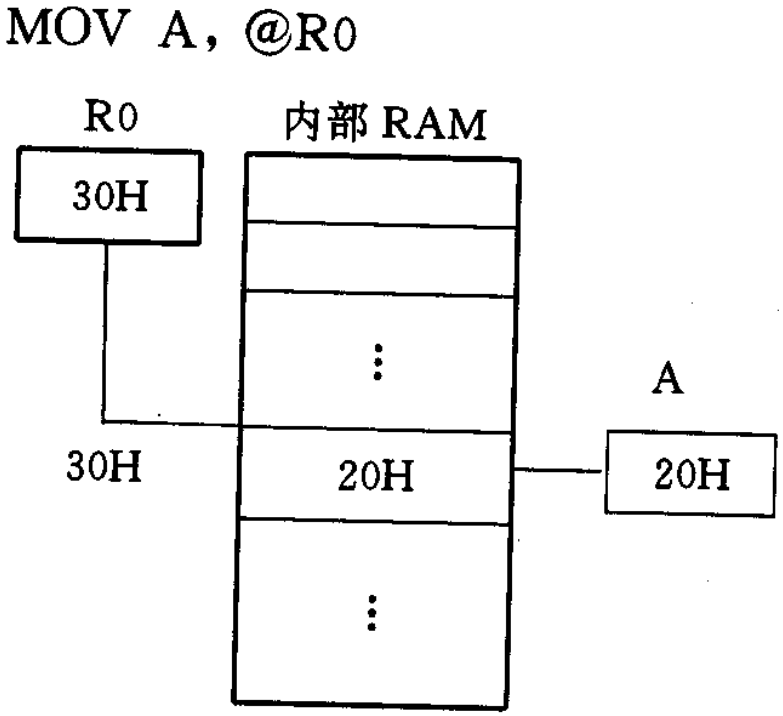
- 在MCS -51 单片机指令系统中, 直接寻址方式中可以访问 2 种存储器空间:
  - (1) 内部数据存储器的低 128 个字节单元（00H～7FH）。
  - (2) 特殊功能寄存器。 特殊功能寄存器只能用直接寻址方式进行访问。（ACC等同于0E0H）

寄存器寻址

- 在该寻址方式中, 参加操作的数存放在寄存器里。寄存器包括8个工作寄存器R0～R7, 累加器A。
- MOV A，Rn ；A←（Rn） 其中n为0~7之一，Rn是工作寄存器。
- ADD A, Rn ；A ← Rn中内容与A相加
- MOV Rn，A ；Rn←（A）

寄存器间接寻址

- 在这种寻址方式中，寄存器的内容为操作数的地址。寄存器间接寻址只能使用寄存器R0、R1 作为地址指针，寻址内部RAM区的数据；当访问外部RAM时，可使用R0、 R1及DPTR作为地址指针。
- 寄存器间接寻址符号为“@”



寄存器间接寻址（2）

例1：

```
MOV   R0，#30H   ；R0←30H
MOV   A，  @R0   ；A ← (R0)
MOV   A ，@R1    ；A ← (R1)
```

例2：

```
MOV DPTR，#3456H ；DPTR←3456H
MOVX  A，@DPTR    ；A ← (DPTR)
```

C语言：

i_pointer=&i；

i=3；

*** i_pointer=3；**

变址寻址

- 用于访问程序存储器中的数据表格, 它以基址寄存器DPTR或PC的内容为基本地址, 加上变址寄存器A的内容作为操作数的地址, 例如:
  - MOVC　A, @A+DPTR　;93H
  - MOVC　A, @A+ PC　　;83H
  - JMP　@A+DPTR　　;73H
- 这三条指令均为单字节指令

相对寻址

- 51转移指令分为直接转移和相对转移指令, 在相对转移指令中采用相对寻址方式。
- 这种寻址方式是以PC的内容为基本地址, 加上指令中给定的偏移量作为转移地址。指令中给出的偏移量是一个 8 位带符号的常数, 可正可负, 其范围为－128～+127。
- **SJMP 08H ; PC←转移指令地址+2+08H**
- 实现C语言中循环和选择语句功能。

位寻址

- 该种寻址方式中，操作数是内部RAM单元中某一位的信息。
  - SETB TR0 ；TR0 ← 1
  - CLR 00H ；(00H) ← 0
  - MOV C，57H ；位寻址

    将57H位地址的内容传送到位累加器C
  - MOV A，57H ；直接寻址

寻址方式和寻址空间

寻址方式	寻址空间
1.立即寻址;	程序存储器
2.直接寻址;	内部RAM低128字节 特殊功能寄存器（SFR）
3.寄存器寻址;	R0~R7，A
4.寄存器间接寻址;	内部RAM(@R0，@R1) 外部RAM(@R0，@R1 ，@DPTR)
5.变址寻址;	程序存储器(@A+DPTR,@A+PC)
6.相对寻址;	程序存储器(PC+偏移量)
7.位寻址。	内部RAM中128个可寻址位 SFR中可寻址位

第三章　MCS-51指令系统

- §3-1　指令系统概述
- §3-2　寻址方式
- §3-3　8051指令集

数据传送类指令

- 实现寄存器、存储器之间的数据传送，用于保存、交换数据。
- 内部数据传送指令
  - 格式： MOV [目的操作数]，[源操作数]
  - 功能：目的操作数 ←（源操作数中的数据）
  - 源操作数可以是：A、Rn、direct、@Ri、#data
  - 目的操作数可以是：A、Rn、direct、@Ri

以累加器A为目的操作数

```
MOV     A，Rn          ；(A)←（Rn）
MOV     A，direct      ；(A)←（direct）
MOV     A，@Ri         ；(A)←（（Ri））
MOV     A，#data       ；(A)← data
```

以Rn为目的操作数

```
MOV     Rn，A              ；(Rn)←（A）

MOV     Rn，direct         ；(Rn)←（direct）

MOV     Rn，#data          ；(Rn)←data
```

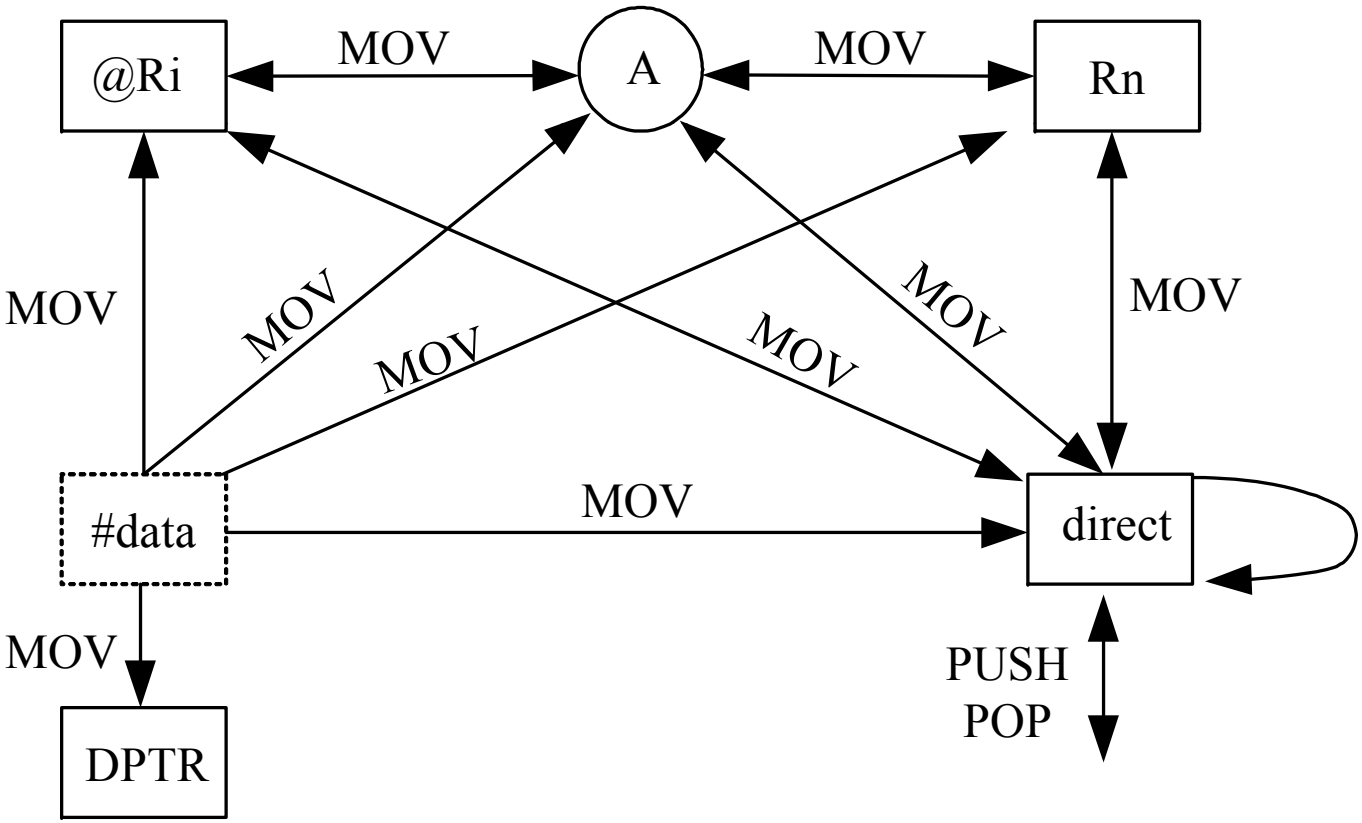
以内部RAM或SFR为目的操作数

```
MOV     direct，A          ；(direct)←(A)
MOV     direct，Rn         ；(direct)←(Rn)
MOV     direct1，direct2   ；(direct) 1 ←(direct2)
MOV     direct，@Ri        ；(direct)←((Ri))
MOV     direct，#data      ；(direct)←#data
MOV     @Ri，A             ；((Ri))←( A )
MOV     @Ri，direct        ；((Ri))←(direct)
MOV     @Ri，#data         ；((Ri))←data
MOV     DPTR，#data16      ；DPTR←data16
```

堆栈操作

- 所谓堆栈是在片内RAM中按“先进后出，后进先出”原则设置的专用存储区。数据的进栈出栈由指针SP统一管理。堆栈的操作有如下两条专用指令：
- PUSH direct；SP←（SP+1），（SP）←（direct）
- POP direct；（direct）←（SP），SP ← SP-1
- PUSH是进栈（或称为压入操作）指令。
- POP是出栈（或称为弹出操作）指令。

片内数据传送指令MOV、PUSH和POP共18条。



字节交换指令

- 数据交换主要是在内部RAM单元与累加器A之间进行，有整字节和半字节两种交换。
- 字节交换
  - **XCH A，Rn ；(A)⇔(Rn)**
  - **XCH A，direct ；(A)⇔(direct)**
  - **XCH A，@Ri ；(A)⇔((Ri))**
- 半字节交换
  - **XCHD A，@Ri ；$(A)_{0\sim3}⇔((Ri))_{0\sim3}$**
  - **SWAP A ；$(A)_{0\sim3}⇔(A)_{4\sim7}$**

【例】 将片内RAM 30H单元与40H单元中的内容互换。

- 方法1（直接地址传送法）：

```
MOV     31H，30H
MOV     30H，40H
MOV     40H，31H
SJMP    $
```

- 方法2（间接地址传送法）：

```
MOV     R0，#40H
MOV     R1，#30H
MOV     A，@R0
MOV     B，@R1
MOV     @R1，A
MOV     @R0，B
SJMP    $
```

- 方法3（字节交换传送法）：

```
MOV     A，30H
XCH     A，40H
MOV     30H，A
SJMP    $
```

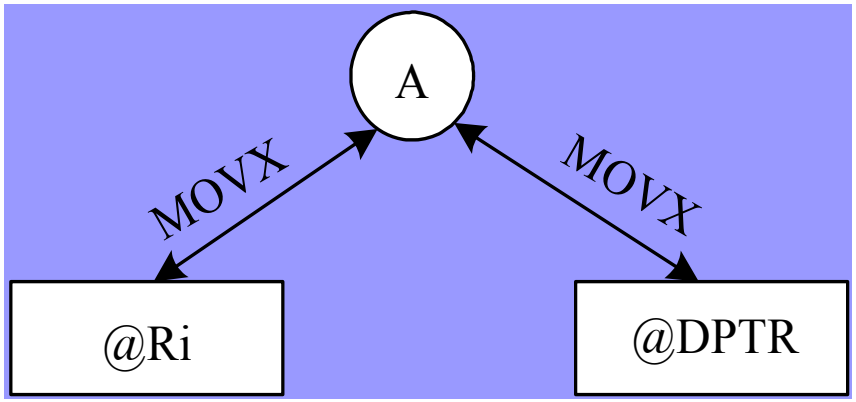
- 方法4（堆栈传送法）：

```
PUSH    30H
PUSH    40H
POP     30H
POP     40H
SJMP    $
```

累加器A与外部数据存储器之间的传送指令

```
MOVX    A，@DPTR        ；(A)←((DPTR))
MOVX    A，@Ri          ；(A)←((Ri))
MOVX    @DPTR，A        ；((DPTR))←(A)
MOVX    @Ri，A          ；((Ri))← (A)
```

片外数据存储器数据传送指令MOVX共4条。

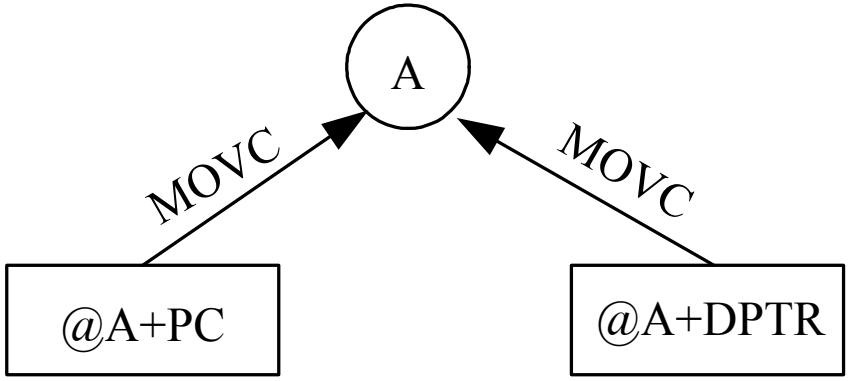


程序存储器内容送累加器

MOVC A，@A+PC

MOVC A，@A+DPTR

程序存储器查表指令MOVC共2条



数据运算类指令

指令助记符	功能简述	字节数	振荡器周期数
ADD A，Rn	A←(A)+(Rn)	1	12
ADD A，direct	A←(A)+(direct)	2	12
ADD A，@Ri	A←((Ri))+(A)	1	12
ADD A，#data	A←(A)+data	2	12
ADDC A，Rn	A←(A)+(Rn)+Cy	1	12
ADDC A，direct	A←(A)+(direct)+Cy	2	12
ADDC A，@Ri	A←(A)+(Ri)+Cy	1	12
ADDC A，#data	A←(A)+data+Cy	2	12
INC A	A←(A)+1	1	12
INC Rn	Rn←(Rn)+1	1	12

数据运算类指令（2）

INC @Ri	(Ri)←((Ri))+1	1	12
INC direct	direct←(direct)+1	2	12
INC DPTR	DPTR←(DPTR)+1	1	24
DA A	对 A 进行十进制调整	1	12
SUBB A，Rn	A←(A)－(Rn)－Cy	1	12
SUBB A，@Ri	A←(A)－(Ri)－Cy	1	12
SUBB A，direct	A←(A)－direct－Cy	2	12
SUBB A，#data	A←(A)－data－Cy	2	12
DEC A	A←(A)－1	1	12
DEC Rn	Rn←(Rn)－1	1	12
DEC direct	direct←(direct)－1	2	12
DEC @Ri	(Ri) ←((Ri))－1	1	12
MUL AB	AB←(A) * (B)	1	48
DIV AB	AB←(A)/(B)	1	48

例：两个16位数加法

- 两个16位数，低8位分别存于20H和30H之中。高八位分别存于21H和31H中。求他们的和，和的低八位送40H，高8位送41H。

```
MOV     A,    20H
ADD     A,    30H    ；低8位相加
MOV     40H,  A      ；存和的低8位
MOV     A,    21H
ADDC    A,    31H    ；高8位相加,带低八位相加进位
MOV     41H,  A      ；存和的高8位
```

影响标志位的指令

指令	标志位		
	Cy	OV	AC
ADD	**X**	**X**	**X**
ADDC	**X**	**X**	**X**
SUBB	**X**	**X**	**X**
MUL	**0**	**X**	
DIV	**0**	**X**	
DA A	**X**		**X**
RRC	**X**		
RLC	**X**		
SETB C	**1**		
CLR C	**0**		
CPL C	**X**		
ANL	**X**		
ORL	**X**		
MOV C, bit	**X**		
CJNE	**X**		

十进制调整指令

- DA A
- 这条指令对累加器A参与的BCD码加法运算所获得的 8 位结果进行十进制调整, 使累加器A中的内容调整为二位压缩型 BCD码的数。
- 使用时必须注意, 它只能跟在加法指令之后, 不能对减法指令的结果进行调整, 且其结果不影响溢出标志位。
- 执行该指令时, 判断A中的低 4 位是否大于 9 和辅助进位标志 AC是否为 “1”, 若两者有一个条件满足, 则低 4 位加 6 操作; 同样, A中的高 4 位大于 9 或进位标志 Cy为 “1”两者有一个条件满足时, 高 4 位加 6 操作。

例如: 有两个 BCD数 36 与 45 相加, 结果应为 BCD码 81

```
MOV  A, #36H
ADD  A, #45H
DA   A
```

这段程序中, 第一条指令将立即数36H(BCD码36)送入累加器A; 第二条指令进行如下加法:

```
  0 0 1 1 0 1 1 0    36
+ 0 1 0 0 0 1 0 1    45
-------------------------
  0 1 1 1 1 0 1 1    7B
+ 0 0 0 0 0 1 1 0    06
-------------------------
  1 0 0 0 0 0 0 1    81
```

得结果 7BH; 第三条指令对累加器 A进行十进制调整, 低 4 位(为 0BH)大于 9, 因此要加 6, 得调整的 BCD码 81。

乘法指令

乘法指令完成单字节的乘法, 只有一条指令:

MUL AB

这条指令的功能是: 将累加器A的内容与寄存器B的内容相乘, 乘积的低 8 位存放在累加器A中, 高 8 位存放于寄存器B中。如果乘积超过0FFH, 则溢出标志OV置“1″, 否则清“0”。 进位标志Cy总是被清“0”。

除法指令

除法指令完成单字节的除法, 只有一条指令:

DIV AB

这条指令的功能是: 将累加器 A中的内容除以寄存器 B中的8位无符号整数, 所得商的整数部分存放在累加器A中, 余数部分存放在寄存器 B中, 清“0″进位标志Cy和溢出标志OV。若原来 B中的内容为 0, 则执行该指令后 A与 B中的内容不定, 并将溢出标志OV置“1”, 在任何情况下, 进位标志Cy总是被清“0”。

例：数的码制转换。把累加器A中无符号二进制整数（00-FFH）转换为三位压缩BCD码（0～255），并存入内存30H和31H单元。

```
BINBCD:   MOV   B，#100
          DIV   AB              ；A÷100位数在A,余数在B
          MOV   30H，A          ；百位数送30H
          MOV   A，B
          MOV   B，#0AH
          DIV   AB              ；余数÷10，十位数在A低四位,
                                ；个位数在B
          SWAP   A              ；十位数放A的高四位
          ADD    A，B           ；十位数和个位数组合后送31H
          MOV    31H，A
          RET
```

逻辑运算类指令

指令助记符	功 能 简 述	字节数	振荡器周期数
CLR A	累加器清零	1	12
CPL A	累加器取反	1	12
RL A	累加器循环左移 1 位	1	12
RLC A	累加器带进位标志位循环左移 1 位	1	12
RR A	累加器循环右移 1 位	1	12
RRC A	累加器带进位标志位循环右移 1 位	1	12
ANL A，Rn	A←(A)∧(Rn)	1	12
ANL A，direct	A←(A)∧(direct)	2	12
ANL A，@Ri	A←(A)∧(Ri)	1	12
ANL A，#data	A←(A)∧data	2	12
ANL direct，A	direct←(direct)∧(A)	2	12
ANL direct，#data	direct←(direct)∧data	3	24
ORL A，Rn	A←(A)∨(Rn)	1	12
ORL A，direct	A←(A)∨(direct)	2	12
ORL A，@Ri	A←(A)∨((Ri))	1	12
ORL A，#data	A←(A)∨data	2	12
ORL direct，A	direct←(direct)∨(A)	2	12
ORL direct，#data	direct←(direct)∨data	3	24
XRL A，Rn	A←(A)⊕(Rn)	1	12
XRL A，direct	A←(A)⊕(direct)	2	12
XRL A，@Ri	A←(A)⊕((Ri))	1	12
XRL A，#data	A←(A)⊕data	2	12
XRL direct，A	direct←(direct)⊕(A)	2	12
XRL direct，#data	direct←(direct)⊕data	3	24

一、 简单逻辑操作指令

CLR A ; 对累加器A清“0″

CPL A; 对累加器A按位取反

RL A; 累加器A的内容向左环移 1 位

RLC A; 累加器A的内容带进位标志位向左环移 1 位

RR A; 累加器A的内容向右环移 1 位

RRC A; 累加器A的内容带进位标志位向右环移 1 位

这组指令的功能是: 对累加器A的内容进行简单的逻辑操作。除了带进位标志位的移位指令外,其它都不影响Cy, AC, OV等标志。

二、 逻辑与指令

ANL A, Rn

ANL A, direct

ANL A, @Ri

ANL A, #data

ANL direct, A

ANL direct, #data

这组指令的功能是：将两个操作数的内容按位进行逻辑与操作, 并将结果送回目的操作数的单元中。

三、 逻辑或指令

ORL A, Rn

ORL A, direct

ORL A, @Ri

ORL A, #data

ORL direct, A

ORL direct, #data

这组指令的功能是: 将两个操作数的内容按位进行逻辑或操作, 并将结果送回目的操作数的单元中。

四、 逻辑异或指令

XRL A, Rn

XRL A, direct

XRL A, @Ri

XRL A, #data

XRL direct, A

XRL direct, #data

这组指令的功能是：将两个操作数的内容按位进行逻辑异或操作, 并将结果送回到目的操作数的单元中。

控制转移类指令

控制转移指令共有 17 条, 不包括按布尔变量控制程序转移指令。其中有 64 KB范围内的长调用、 长转移指令; 有 2 KB范围内的绝对调用和绝对转移指令; 有全空间的长相对转移及一页范围内的短相对转移指令; 还有多种条件转移指令。由于MCS -51 提供了较丰富的控制转移指令, 因此在编程上相当灵活方便。这类指令用到的助记符共有 10 种: AJMP、LJMP、SJMP、JMP、ACALL、LCALL、JZ、JNZ、CJNE、DJNZ。

控制转移指令

指令助记符	功能简述	字节数	振荡器周期数
AJMP $addr_{11}$	2 KB 内绝对转移	2	24
LJMP $addr_{16}$	64 KB 内绝对转移	3	24
SJMP rel	相对短转移	2	24
JMP @A+DPTR	相对长转移	1	24
JZ rel	累加器为零转移	2	24
JNZ rel	累加器不为零转移	2	24
CJNE A, direct, rel	A 的内容与直接寻址字节的内容不等转移	3	24
CJNE A, #data, rel	A 的内容与立即数不等转移	3	24
CJNE Rn, #data, rel	Rn 的内容与立即数不等转移	3	24
CJNE @Rn , #data , rel	内部 RAM 单元的内容与立即数不等转移	3	24
DJNZ Rn, rel	寄存器内容减 1不为零转移	2	24
DJNZ direct, rel	直接寻址字节内容减1不为零转移	3	24
ACALL $addr_{11}$	2 KB 内绝对调用	2	24
LCALL $addr_{16}$	64 KB 内绝对调用	3	24
RET	子程序返回	1	24
RETI	中断返回	1	24

一、无条件转移指令

1. 绝对转移指令

AJMP $addr_{11}$

这是2KB范围内的无条件跳转指令, 执行该指令时, 先将PC+2, 然后将$addr_{11}$送入PC_{10}～PC_0, 而PC_{15}～PC_{11}保持不变。这样得到跳转的目的地址。需要注意的是, 目标地址与AJMP后面一条指令的第一个字节必须在同一个 2 KB区域的存储器区内。

2. （短）相对转移指令

SJMP rel

执行该指令时, 先将 PC+2, 再把指令中带符号的偏移量加到PC上, 得到跳转的目标地址送入PC。 rel为有符号8位二进制数。

3. 长跳转指令

LJMP addr16

执行该指令时, 将 16 位目标地址addr16 装入PC, 程序无条件转向指定的目标地址。转移的目标地址可以在 64 KB程序存储器地址空间的任何地方, 不影响任何标志。

4. 散转指令

JMP @A+DPTR

执行该指令时, 把累加器 A中的 8 位无符号数与数据指针中的 16 位数相加, 结果作为下条指令的地址送入PC, 不改变累加器 A和数据指针DPTR的内容, 也不影响标志。 利用这条指令能实现程序的散转。

例：switch程序设计，根据A中的数值实现程序跳转

```
        MOV     R1,A                ；(A)×3
        RL      A
        ADD     A,R1
        MOV     DPTR，  #TABLE   ；散转表首地址送DPTR
        JMP     @A+DPTR
TABLE：  LJMP    PM0                 ；转程序PM0
TABLE＋3：LJMP   PM1                 ；转程序PM1
          ------
PM0：    -----
```

LJMP是一个三字节指令，因此转移指令入口地址相隔3个字节，A中内容需是3的倍数。

二、 条件转移指令

JZ rel ；（A）= 0 转移

JNZ rel ；（A）≠0 转移

这类指令是依据累加器A的内容是否为 0 的条件转移指令。条件满足时转移（相当于一条相对转移指令），条件不满足时则顺序执行下面一条指令。转移的目标地址在以下一条指令的起始地址为中心的 256 个字节范围之内（－128～ +127）。当条件满足时, PC←(PC)+N+rel, 其中(PC)为该条件转移指令的第一个字节的地址, N为该转移指令的字节数（长度）, 本转移指令N=2。

三、 比较转移指令

在MCS - 51 中没有专门的比较指令, 但提供了下面 4 条比较不相等转移指令:

CJNE A, direct, rel

CJNE A, #data, rel

CJNE Rn, #data, rel

CJNE @Ri, #data, rel

这组指令的功能是: 比较前面两个操作数的大小, 如果它们的值不相等则转移。转移地址的计算方法与上述两条指令相同。如果第一个操作数（无符号整数）小于第二个操作数, 则进位标志Cy置“1”, 否则清“0”, 但不影响任何操作数的内容。

JC rel; 若(Cy)=1, 则转移 PC←(PC)+2+rel

JNC rel; 若(Cy)=0, 则转移PC←(PC)+2+rel

例：温度控制程序

某温度控制系统，A中存温度采样值Ta，(20H)=温度下限值T20，(30H)=温度上限值T30。若Ta＞T30，程序转降温JW，若Ta＜T20，程序转升温SW，若T30≥Ta≥T20程序转FH返回主程序。

```
        CJNE  A，30H ，  LOOP
        AJMP  FH                    ；等于T30，转FH
LOOP：  JNC  JW                     ；大于T30，降温
        CJNE  A，20 H，LOOP1
        AJMP  FH                    ；等于T20，转FH
LOOP1： JC  SW                      ；小于T20，升温
  FH：  ------                      ；保温
  JW：  ------
  SW：  ------
```

四、减 1不为 0转移指令

DJNZ Rn, rel

DJNZ direct, rel

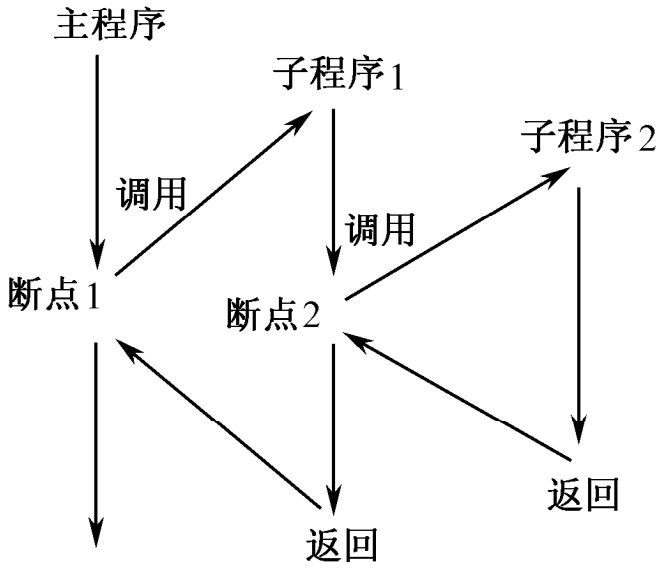
这两条指令把源操作数减 1, 结果回送到源操作数中去, 如果结果不为 0 则转移（转移地址的计算方法同前）。

例：从P1.7引脚输出5个方波

```
        MOV  R2，#10      ；5个方波，10个状态
LOOP：  CPL  P1.7         ；P1.7状态变反
        DJNZ  R2 ，LOOP
```

五、调用及返回指令

在程序设计中，通常把具有一定功能的公用程序段编制成子程序，当主程序需要使用子程序时用调用指令，而在子程序的最后安排一条子程序返回指令,以便执行完子程序后能返回主程序继续执行。



1. 绝对调用指令

ACALL $addr_{11}$

这是一条 2 KB范围内的子程序调用指令。执行该指令时，先将 PC+2 以获得下一条指令的地址, 然后将 16 位地址压入堆栈（PCL内容先进栈, PCH内容后进栈）, SP内容加2, 最后把 PC的高 5 位PC_{15}～PC_{11}与指令中提供的 11 位地址$addr_{11}$相连接（PC_{15}～PC_{11}，A_{10}～A_0），形成子程序的入口地址送入PC, 使程序转向子程序执行。所用的子程序的入口地址必须与 ACALL下面一条指令的第一个字节在同一个2 KB区域的存储器区内。

2. 长调用指令

LCALL $addr_{16}$

这条指令无条件调用位于 16 位地址$addr_{16}$的子程序。执行该指令时，先将PC+3以获得下一条指令的首地址，并把它压入堆栈（先低字节后高字节），SP内容加 2，然后将 16 位地址放入 PC中，转去执行以该地址为入口的程序。 LCALL指令可以调用 64 KB范围内任何地方的子程序。指令执行后不影响任何标志。

3. 子程序返回指令

RET

这条指令的功能是: 恢复断点, 将调用子程序时压入堆栈的下一条指令的首地址取出送入PC, 使程序返回主程序继续执行。

4. 中断返回指令

RETI

这条指令的功能与RET指令相似, 不同的是它还要清除MCS -51 单片机内部的中断状态标志。

位操作类指令

指令助记符	功能简述	字节数	振荡器周期数
MOV C，bit	Cy ←(bit)	2	12
MOV bit，C	bit← Cy	2	12
CLR C	Cy←0	1	12
CLR bit	bit ←0	2	12
CPL C	Cy←($\overline{Cy}$)	1	12
CPL bit	bit←($\overline{bit}$)	2	12
SETB C	Cy← 1	1	12
SETB bit	bit← 1	2	12
ANL C，bit	Cy←(Cy) ∧ (bit)	2	24
ANL C，/bit	Cy←(Cy) ∧ ($\overline{bit}$)	2	24
ORL C，bit	Cy←(Cy) ∨ (bit)	2	24
ORL C，/bit	Cy←(Cy) ∨ ($\overline{bit}$)	2	24
JC rel	若(Cy)＝1，则转移，PC←(PC)＋2＋rel	2	24
JNC rel	若(Cy)＝0，则转移，PC←(PC)＋2＋rel	2	24
JB bit，rel	若(bit)＝1，则转移，PC←(PC)＋3＋rel	3	24
JNB bit，rel	若(bit)＝0，则转移，PC←(PC)＋3＋rel	3	24
JBC bit，rel	若(bit)＝1，则转移，PC←(PC)＋3＋rel，并 bit←0	3	24

1. 位数据传送指令

MOV C, bit

MOV bit, C

这组指令的功能是：把源操作数指出的布尔变量送到目的操作数指定的位地址单元中。其中一个操作数必须为进位标志 Cy, 另一个操作数可以是任何可直接寻址位。

2. 位变量修改指令

CLR C

CLR bit

CPL C

CPL bit

SETB C

SETB bit

这组指令对操作数所指出的位进行清“0”，取反，置“1”的操作，不影响其它标志。

3. 位变量逻辑与指令

ANL C, bit

ANL C, /bit

这组指令的功能是: 如果源位的布尔值是逻辑 0, 则将进位标志清“0”; 否则, 进位标志保持不变, 不影响其它标志。bit前的斜杠表示对(bit)取反, 直接寻址位取反后用作源操作数, 但不改变直接寻址位原来的值。例如指令: ANL C, /ACC.0 执行前ACC.0 为 0, C为 1, 则指令执行后 C为 1, 而 ACC.0仍为 0。

4. 位变量逻辑或指令

ORL C, bit

ORL C, /bit

这组指令的功能是: 如果源位的布尔值是逻辑 1, 则将进位标志置“1”; 否则, 进位标志保持不变, 不影响其它标志。

5. 位变量条件转移指令

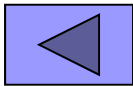
JC rel; 若(Cy)=1, 则转移 PC←(PC)+2+rel

JNC rel; 若(Cy)=0, 则转移PC←(PC)+2+rel

JB bit, rel; 若(bit)=1, 则转移PC←(PC)+3+rel

JNB bit, rel; 若(bit)=0, 则转移PC←(PC)+3+rel

JBC bit, rel; 若(bit)=1, 则转移PC←(PC)+3+rel, 并 bit← 0



思考题

- 简述８０C５１的指令寻址方式，并举例说明。
- 访问特殊功能寄存器SFR，可使用哪些寻址方式？
- 若访问外部RAM单元，可使用哪些寻址方式？
- 若访问内部RAM单元，可使用哪些寻址方式？
- 若访问程序存储器，可使用哪些寻址方式？
- MOV、MOVC、MOVX指令有什么区别？分别用于哪些场合？为什么？
- 说明“DA A”指令功能，并说明二－十进制调整的原理和方法。